Machine Learning Applied to Bike Sharing Demand Data

Article citation: E, S., Park, J. and Cho, Y., 2020. Using data mining techniques for bike sharing demand prediction in metropolitan city. Computer Communications, 153, pp.353-366.

Introduction and objective

Shared bicycle rental schemes are increasingly used in many urban cities as a primary mode of transportation. For this to be effective, it is essential that rental bikes be available at the times and locations that they are required. In a recent research article published in the journal Computer Communications, authors Sathishkumar V E, Jangwoo Park, and Yongyun Cho sought to predict the "bike count required at each hour for the stable supply of rental bikes"[1]. They considered a dataset that includes weather information (Temperature, Humidity, Windspeed, Visibility, Dewpoint, Solar radiation, Snowfall, Rainfall), the number of bikes rented per hour and date information. They employed a number of regression models, including linear regression. In this assignment we will reproduce some of the results of Ref. [1] on a slightly modified version of the original dataset.

[1] Sathishkumar V E, Jangwoo Park, and Yongyun Cho. 'Using data mining techniques for bike sharing demand prediction in metropolitan city.' Computer Communications, Vol.153, pp.353-366, March, 2020. web link.

Read the Data

In [1]:

```
# (1) Import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from sklearn import datasets
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.tree import DecisionTreeRegressor
```

We are going to work with an adapted Database "SeoulBikeData_mod.csv", call it SBD (initials). You can see this data below.

In [2]:

```
# (2)(3) read the SeoulBikeData_mod.csv file to dataframe SBD and describe it to check.
SBD = pd.read_csv("SeoulBikeData_mod.csv")
SBD.describe()
```

Out[2]:

	Rented Bike Count	Hour	Temperature(C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(C)	Solar Radiation (MJ/m2)	Rainfa
count	4220.000000	4220.000000	4220.000000	4220.000000	4220.000000	4220.000000	4220.000000	4220.000000	4220.0
mean	665.802607	11.522512	9.476967	54.729147	1.883886	1342.977725	-0.066256	0.606507	0.0
std	652.252686	6.930339	11.186973	20.994968	1.084647	622.576774	12.440532	0.910329	0.9
min	20.000000	0.000000	-17.800000	0.000000	0.000000	27.000000	-30.600000	0.000000	0.0
25%	177.000000	6.000000	1.100000	38.000000	1.000000	770.750000	-9.800000	0.000000	0.0
50%	397.000000	12.000000	10.000000	53.000000	1.700000	1495.000000	0.600000	0.020000	0.0

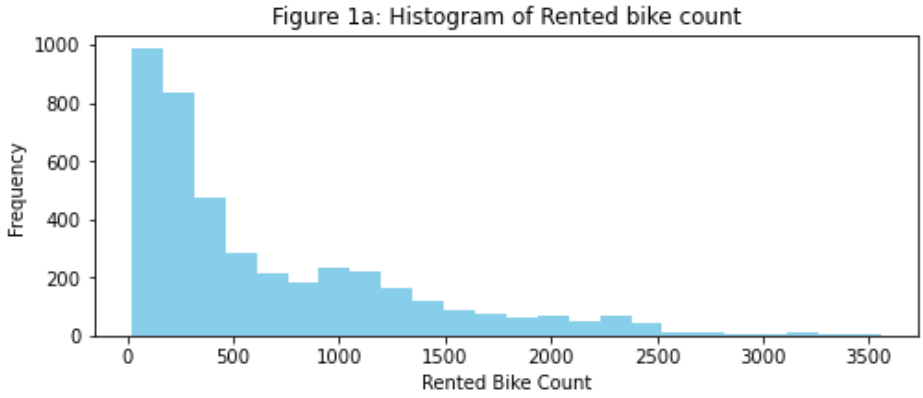
	Rented Bike Count	Hour	Temperature(C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point temperature(C)	Solar Radiation (MJ/m2)	Rainfa
75%	1020.250000	18.000000	19.100000	70.000000	2.600000	1972.000000	10.425000	0.992500	0.0
max	3556.000000	23.000000	32.700000	98.000000	7.400000	2000.000000	24.100000	3.520000	35.0

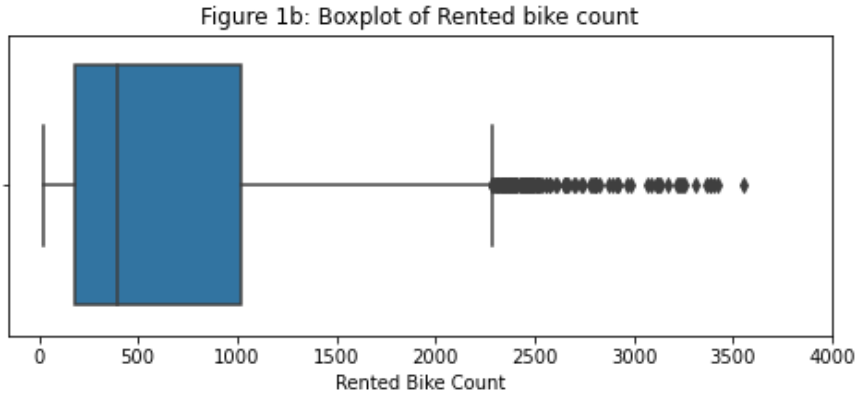
This histogram allows us to illustrate the data represented in the above table. We can see that the "closer to" 4000 the Rented Bike Count is the frequency approaches 0. Although this values of frequency depend on the number of bins.

Visualizing the data

In [3]:

```
# (4) Plot histogram of the Rented bike count.
# The number of bins is selected as 24 for resemblance with the article.
fig = plt.figure(figsize=(8,3))
test_series = SBD["Rented Bike Count"]
plt.hist(test_series, bins=24, color = "skyblue")
plt.xlabel("Rented Bike Count")
plt.ylabel("Frequency")
plt.title("Figure 1a: Histogram of Rented bike count")
plt.show()

# (4)Bonus. Boxplot plotting.
fig = plt.figure(figsize=(8,3))
plt.title("Figure 1b: Boxplot of Rented bike count")
sns.boxplot( x=SBD["Rented Bike Count"] );
plt.xlim(right=4000)
plt.show()
```



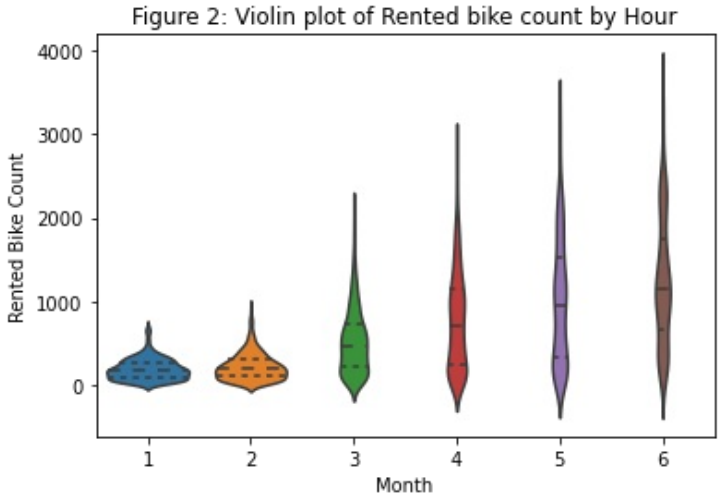


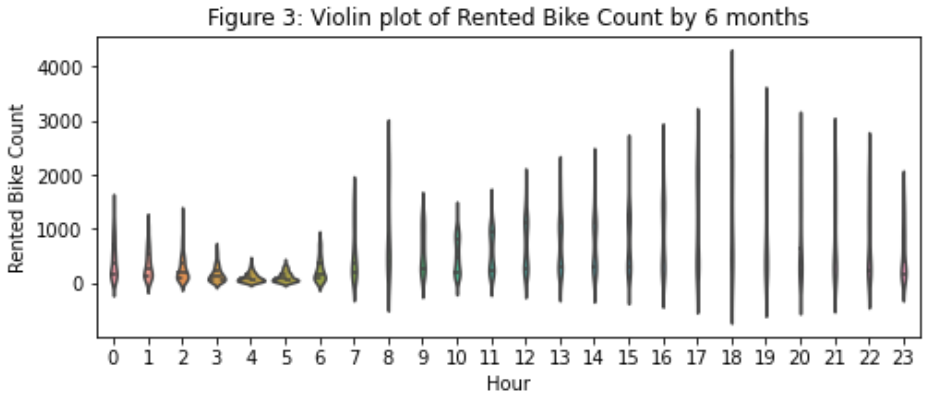
Violin plots are also very illustrative.

In [4]:

```
# (5) Violin plots per Month and per Hour.
# We can observe that the violin plots per hour are much more narrow
# This makes sense as there are less bikes rented in an hour than in a month.
sns.violinplot(x='Month', y='Rented Bike Count', data=SBD, inner='quartile')
plt.title("Figure 2: Violin plot of Rented bike count by Hour")
plt.show()
```

```
fig = plt.figure(figsize=(8,3))
sns.violinplot(x='Hour', y='Rented Bike Count', data=SBD, inner='quartile')
plt.title("Figure 3: Violin plot of Rented Bike Count by 6 months")
plt.show()
```





Preparing the data

First we extract the 'Rented Bike Count' column as a target vector `y`. Then we construct the design matrix `X` as all columns except the 'Rented Bike Count'. From these we perform a train-test split in which 25% of the samples are saved for testing. This is the same number as was used in Ref. [1].

In [5]:

```
#(6)
# We create our matrix and target.
X = SBD.drop(['Rented Bike Count'], axis=1)
y = SBD['Rented Bike Count']
print(type(X))
print(type(y))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

Perform machine learning

We set up a linear regression learning.

In [6]:

```
#(7)-(9)
# Create a train-test split with a test size of 20% of the total samples
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25)

# Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the train data
```

```
regr.fit(X_train, y_train)

# Use the model to predict the test data
y_pred = regr.predict(X_test)
resid = y_test - y_pred
```

Quantitative test

In [7]:

```
#(10)
# We evaluate the model using permformance measures.
#The values for the three error measures are very similar to the ones found in the articl
e

RMSE = np.sqrt(np.mean(resid**2))
print("The value of RMSE for our prediction is RMSE =", RMSE)

MAE = np.mean(np.abs(resid))
print("The value of MAE for our prediction is MAE =", MAE)

# using scikit-learn to compute R2 (note that r2_score was imported above)
R2 = r2_score(y_test, y_pred)
print("The value of R^2 for our prediction is R^2 =", R2)
```

```
The value of RMSE for our prediction is RMSE = 416.65138523486416
The value of MAE for our prediction is MAE = 304.4258028406826
The value of R^2 for our prediction is R^2 = 0.6105423681220141
```

Plots

Next we show plots summarising the accuracy. The first plot shows residual as a function of the Rented Bike Count. Colours indicate the month. One see that the residuals are smaller in the winter (e.g. Jan and Feb) compare with the late-spring, early summer (e.g. May and June).

Next is plotting the histogram of rented bike count both for the original test data (true) and for the predictions (predicted). It is clear that the prediction do not agree well with the true histogram.

The final two plot show the residual as a function of month and of hour. One sees directly that the residual is higher in months 5 and 6 (May and June) compared with months 1 and 2 (Jan and Feb). One also sees that the residual is higher during rush periods in the morning and in the evening.

In [8]:

```
# (11)

# Relation between residual and Rented Bike count
# We observe that there is a positive relationship between the independent and dependent
variable.
# although we have less sample points for bigger values of Rented Bike count (less precis
ion).

plt.scatter(y_test, resid, color = "skyblue")
plt.xlabel("Rented Bike Count", fontsize="14")
plt.ylabel("residual", fontsize="14")
plt.title("Figure 4: Dependence of residual on target", fontsize="16")
plt.show()

# Generate and plot histograms for the linear prediction and real values.
plt.hist(y_pred, bins=34, color = "gold")
plt.hist(y_test, bins=34, color = 'skyblue')
plt.legend(["y_pred", "y_test"])
plt.xlabel("Rented Bike Count")
plt.ylabel("Frequency")
plt.title("Figure 5: Combined histogram comparing y_pred and y_test")
plt.show()
```

```
# Dpendence of residual on target.
# The size of the scatter plots are proportional to the absolute value of the resid
# The absolute value is chosen to avoid bugs with negative sizes and
# because what we want to ilustrate with the size is the module of the error/resid not th
e sign
# This helps ilustrate the meaning of the y variable.

plt.scatter(X_test['Month'], resid, c = resid , s = np.abs(resid) * 0.15 , cmap = "YlGnB
u")
plt.xlabel("Rented Bike Count", fontsize="14")
plt.ylabel("residual", fontsize="14")
plt.title("Figure 4: Dependence of residual on target", fontsize="16")
plt.show()


plt.scatter(X_test['Hour'], resid ,c = resid, s= np.abs(resid) * 0.15 , cmap = "YlGnBu")
plt.xlabel("Rented Bike Count", fontsize="14")
plt.ylabel("residual", fontsize="14")
plt.title("Figure 4: Dependence of residual on target", fontsize="16")
plt.show()
```

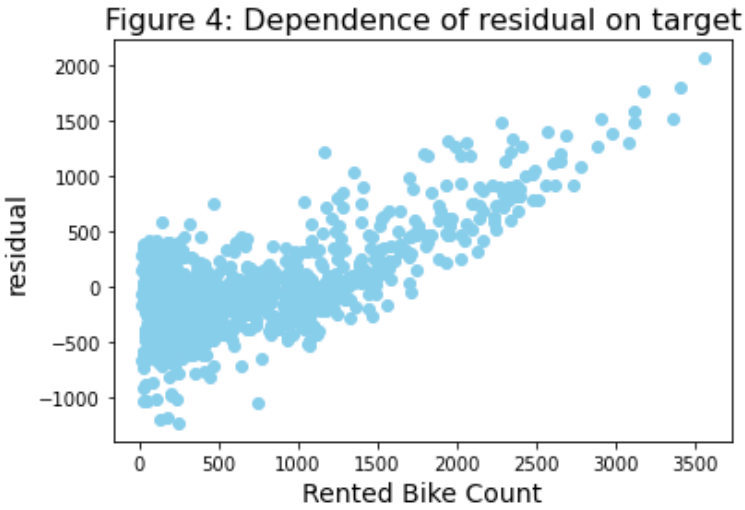
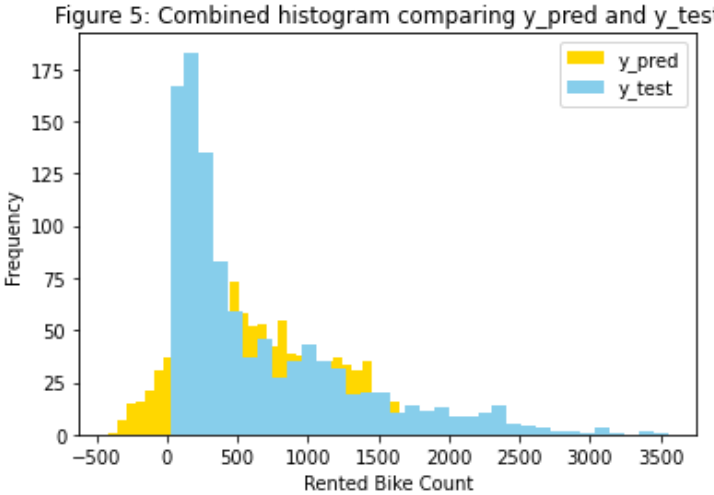


Figure 5: Combined histogram comparing y_pred and y_test



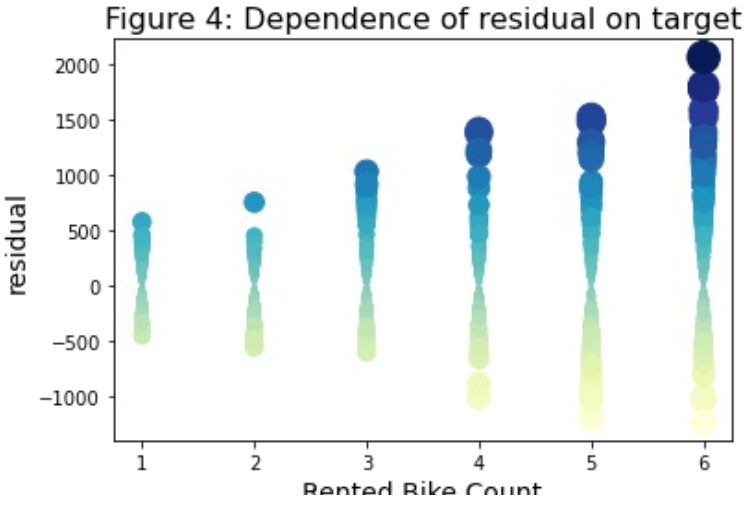
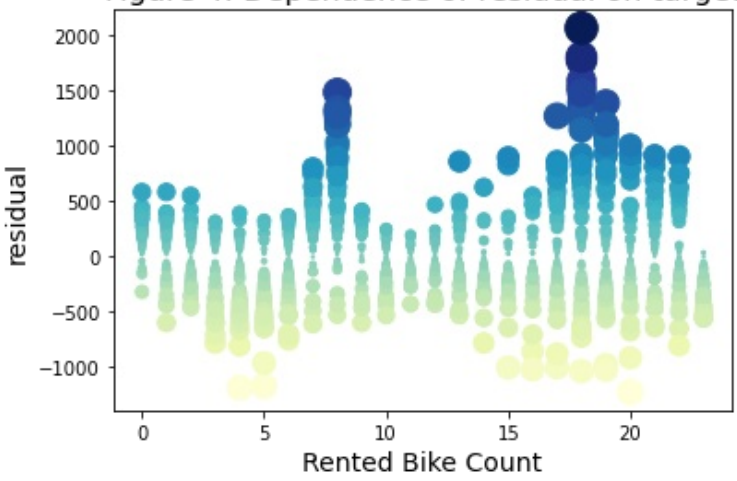


Figure 4: Dependence of residual on target



Decision Tree

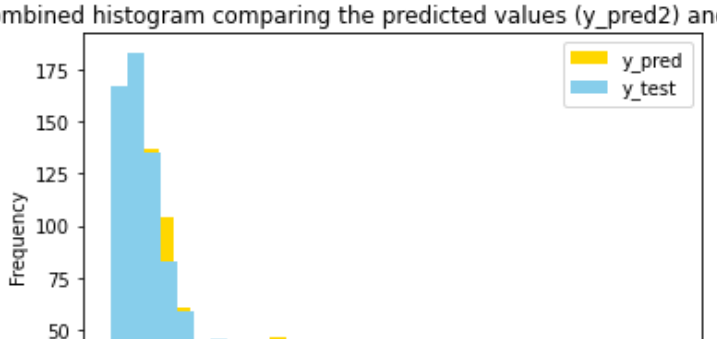
Here we consider a the scikit-learn decision tree regressor. We use the same train-test split from above and train the regressor with a max depth `max_depth=6`.

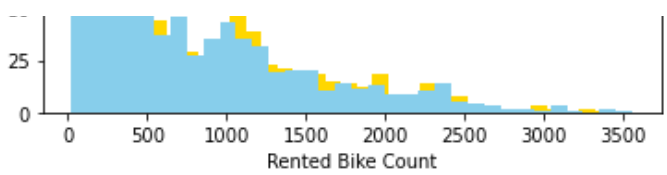
In [9]:

```
# (12) Bonus.
# Fit regression model

regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)
y_pred2 = regressor.predict(X_test)
resid2= y_test - y_pred2



RMSE = np.sqrt(np.mean(resid2**2))
print("The value of RMSE for our DTR prediction is RMSE =", RMSE)

MAE = np.mean(np.abs(resid2))
print("The value of MAE for our DTR prediction is MAE =", MAE)

# using scikit-learn to compute R2 (note that r2_score was imported above)
R2 = r2_score(y_test, y_pred2)
print("The value of R^2 for our DTR prediction is R^2 =", R2)

plt.hist(y_pred2, bins=34, color = "gold")
plt.hist(y_test, bins=34, color = 'skyblue')
plt.legend(["y_pred", "y_test"])
plt.xlabel("Rented Bike Count")
plt.ylabel("Frequency")
plt.title("Figure 5: Combined histogram comparing the predicted values (y_pred2) and the
real (y_test)")
plt.show()
```

```
The value of RMSE for our DTR prediction is RMSE = 317.28581162088574
The value of MAE for our DTR prediction is MAE = 187.52037914691942
The value of R^2 for our DTR prediction is R^2 = 0.774152201164958
```

Figure 5: Combined histogram comparing the predicted values (y_pred2) and the real (y_test)





D) Conclusion and comparison

We have performed machine learning on rented bike count data. We have reproduced some key results from the study of Ref. [1]. In particular, for the linear regression we find values for the Root Mean Squared Error (RMSE), the Mean Absolute Error (MAE), and R squared (R2) similar to those reported in Ref. [1]. We obtain the same behaviour of the residual error rented bike count as did Ref. [1]. We have directly compared histograms of true rented bike counts with predicted values for both linear regression and a decision tree regressor. The decision tree regressor performs considerably better than the linear regressor.

In [ ]: